Zoea

Research note 35

Accelerating Zoea Regression Testing

Edward McDaid & Sarah McDaid
9 Nov 2024

A simple expedient can speed up Zoea regression testing by up to two orders of magnitude.

As most coders are all too aware, whenever a piece of software is modified, defects can crop up elsewhere in their codebase. This can be as a result of dependencies of different kinds that exist in virtually all non-trivial code. Regression testing is intended to detect bugs that occur or reoccur in previously written code, as a result of change. This involves re-running some or all existing functional and non-functional tests, to ensure the correct operation of all components. Regression testing should be conducted at least frequently and increasingly it is carried out continuously.

Zoea employs a large set of regression tests that ensure the correct functioning of all parts of the system during development. The complete suite takes a considerable amount of time and resources to run. Unfortunately, this means that there can be a delay of up to a few hours between a code change and the detection of a regression. Clearly, it would be very helpful to reduce the amount of time and resources required to regression test Zoea.

For a given problem, Zoea can produce many different candidate solutions. By definition, every candidate solution already passes all of the test cases in the program specification. From these, the candidate with the lowest complexity is selected as the canonical solution to the problem. Sometimes, Zoea can produce a number of different but functionally equivalent solutions for the same problem. This can happen where the complexity of candidates is the same but the order in which they are generated varies - often due to the asynchronous nature of the Zoea architecture. In the Zoea ecosystem, having multiple correct solutions is not considered to be an issue. Indeed, one of the key principles upon which the design of Zoea is based is that any solution which passes all of the test cases is a correct solution. As a result, all Zoea regression tests can have a corresponding set of correct results, which may be a single solution or several different (but equivalent) ones.

The Zoea codebase is mostly composed of a large set of components called knowledge sources. Each knowledge source represents a different piece of expertise about programming languages and coding. For example, there are many knowledge sources that represent logical conditions, control structures, operations on collections and so on. In Zoea, solutions are produced by breaking a problem down into progressively smaller pieces until each sub-problem can be solved by a single knowledge source. These solution fragments are then reassembled in different ways and tested until a complete solution can be found, that passes all of the test cases. It is important to note that while there are a large number of Zoea knowledge sources, each solution is produced using only a very small number of them. This fact turns out to be the key to speeding up regression testing in Zoea.

Normally, Zoea can utilise any combination of knowledge sources in order to attempt to solve a problem. However, each knowledge source can also be enabled or disabled independently. Therefore, any desired combination of knowledge sources can be utilised. This facility is invaluable for testing and debugging of knowledge sources during development. The same mechanism can also be used to dramatically reduce the time and effort required to run each regression test. The way this works is as follows.

Take for example any problem with a single known solution. Assume that we know both the solution code for the problem and also the set of knowledge sources which produced it. (This information is readily available to Zoea when each solution is produced.) If we enable only those knowledge sources that are required for the specific problem and disable all others then - assuming everything is working correctly - we will always produce the identical solution whenever we attempt to solve the same problem again. What's more, we will also solve the problem very much more quickly than we would have if all of the knowledge sources had been enabled.

This is because we do not have to consider all of the possible partial solutions produced by other knowledge sources, which never lead to a complete solution. In effect, we have excluded many dead ends that represent wasted time and effort. Furthermore, this approach also excludes complete solutions that are more complex than they need to be, since they will also likely use a larger set of knowledge sources than we have made available. While this latter category do indeed represent valid solutions, the existence of simpler, functionally equivalent solutions makes them redundant. In this sense, they are also a waste of time and effort. Typically, a regression test using this approach will complete up to two orders of magnitude more quickly than one where all knowledge sources are enabled.

This approach also works where the problem can have multiple correct solutions. Here we have to remember and enable the superset of knowledge sources required to produce all of the solutions we want to accept. In such cases, there is often considerable overlap between the sets of knowledge sources required for each alternative solution. Note that where a knowledge source depends on one or more other knowledge sources then all of the knowledge sources must be included in the subset. In any event, the size of the restricted set is still invariably tiny when compared with the total number of knowledge sources.

We can also record the duration required to produce a solution for each regression test, using its restricted set of knowledge sources. This allows us to implement a shorter timeout for the regression test than would be the case if using all knowledge sources. In this way, we can consider how quickly a solution is produced as part of the criteria for regression test success or failure. For example, if a solution is not found within the shorter time frame then the regression test could be rerun with all knowledge sources enabled and with the default timeout.

Based in the above description, we can identify five possible outcomes when running a regression test with a restricted set of knowledge sources enabled:

I. One of the expected solutions is found using the restricted set within the shorter time frame;
II. An unexpected solution is found using the restricted set within the shorter time frame;
III. No solution is found using the restricted set but one of the expected solutions is found using the full set within the default time frame;
IV. No solution is found using the restricted set but an unexpected solution is found using the full set within the default time frame;
V. No solution is found using either the restricted set or the full set.

Finding an expected result is the normal case for most regression testing and with this approach will also be the fastest outcome. At the other end of the spectrum, failure to find any solution will certainly be treated as a failure. How the other cases are interpreted will depend on how the regression tests are being used. For example, an unexpected result might either be a good or bad thing, depending on the circumstances.

Zoea regression tests already include and recognise the code for any number of expected solutions. With our new approach we also need to store the identifiers of the knowledge sources that must be enabled as well as the expected durations, both for the restricted set and for all knowledge sources.

The key concept in this approach is that we have a specific set of knowledge sources that must be remembered and enabled for each regression test, while all others are disabled. There are some obvious parallels between this technique and the instruction subset approach. This is no accident as the current approach was heavily influenced by our recent work on heuristics. The broader application of similar heuristics to knowledge sources (outside of regression testing) is currently being investigated.

The ability to speed up regression testing in this way will make development faster and easier. It enables regression testing to be carried out more frequently and gives developers more timely notice of issues, causing them to have less impact and consequently making them simpler to fix. Indeed, it enables the complete Zoea regression suite to be run on-demand or even continuously. It also helps to reduce the on-going cost of testing, which can be considerable.

Learn more at **zoea.co.uk**